Modular Routing Protocol

MRP

User manual

Rev. L

Nevion
Nordre Kullerød 1
3241 Sandefjord
Norway
Tel: +47 33 48 99 99
nevion.com

Nevion Support

Nevion Europe

P.O. Box 1020
3204 Sandefjord, Norway
Support phone 1: +47 33 48 99 97
Support phone 2: +47 90 60 99 99

Nevion USA

1600 Emerson Avenue
Oxnard, CA 93033, USA
Toll free North America: (866) 515-0811
Outside North America: +1 (805) 247-8560

E-mail: support@nevion.com
See http://www.nevion.com/support/ for service hours for customer support globally.

Revision history

Current revision of this document is the uppermost in the table below.

Rev.	Repl.	Date	Sign	Change description
L	K	2013-10-22	PN/JGS	Added alarms to protocol. Updated events. Added request partial status for parameters. Updated supported commands table, appendix A.
K	J	2013-05-03	JGS	Updated filter command, added chapters for description commands.
J	8	2013-02-04	JGS	Updated supported commands table in appendix
8	7	2012-05-04	JGS	Added description of SI-command, chapter 6.6 Request status – partial status
7	6	2011-01-24	PN	Updated to reflect Multicon release 3.5.0
6	5	2009-12-07	JGS	Updated to reflect Multicon release 3.2.0
5	4	2009-06-02	JIH	First official release
4	3	2009-04-03	JIH	Updated with new style
3	2	2009-04-03	JAP	Added salvos and virtual tables to protocol. Took out deprecated commands. General clean-up.
2	1	2005-10-05	NBS	Command- and response-syntaxes updated.
1	0	2005-07-01	NBS	Some commands updated and more commands added.
0	-	2003-10-08	NBS	Initial release.

Contents

Revision history 2
1 Introduction 5
2 Definitions 6
2.1 Routers and levels 6
2.2 Clients and servers 6
2.3 Message format and structure 6
2.4 Common syntax definitions 7
2.5 Locking 7
2.6 Check Sum (Cyclic Redundancy Checksum, CRC) 8
2.7 Protocol error handling 8
2.8 Future expansion 8
3 Initial communication 9
3.1 Connection to server 9
3.2 User identification and privileges 9
3.3 User list 9
4 Protocol configuration 10
4.1 Protocol syntax 10
4.2 CRC in response 10
4.3 Command echo 11
4.4 Protocol version 11
4.5 Server Ping 11
4.6 Status message suppression 11
5 Logical system structure 13
5.1 Level list 13
5.2 Virtual Table list 13
5.3 Virtual Level list 14
5.4 Parameter group list 14
5.5 Salvo group list 14
5.6 Level input information list 15
5.7 Level output information list 15
5.8 Virtual sources information list 16
5.9 Virtual destinations information list 17
5.10 Category list 17
5.11 Salvo list 17
5.12 Listing available parameters 18
5.13 Changing the alias of an item 19
5.14 Changing the name of an item 20
5.15 Changing the description of an item 20
6 System status 22
6.1 Status messages 22
6.2 Request Crosspoint Status — Entire Level 22
6.3 Request Crosspoint Status – Entire Level – Compact 23
6.4 Request Virtual Crosspoint Status — Entire Virtual Table 23
6.5 Request Salvo Status — Entire Salvo Group 24
6.6 Request status – partial status 24
6.6.1 Levels 24

6.6.2 Virtual router....25
6.6.3 Salvos....25
6.6.4 Parameters....26
6.7 Input signal presence status....27
6.8 Output signal presence status....28
6.9 Getting status for all parameters in group....29
7 System operation commands....30
7.1 Crosspoint Take....30
7.2 Order of execution....30
7.3 Virtual Crosspoint Take....31
7.4 Salvo Take....31
7.5 Setting parameter....32
7.6 Lock/protect....32
7.7 Reset lock/protect....32
7.8 Diagonal....32
8 Server Alarms and Events....33
8.1 Alarm status....33
8.2 Alarm list....34
8.3 Alarm acknowledge....34
8.4 Event status....35
9 Deprecated commands....36
9.1 Request Crosspoint Locking Status — Single Destination....36
9.2 Router list....36
9.3 Router partition list....36
9.4 Bus list....37
9.5 Module list....37
9.6 List input configuration and status....37
9.7 List output configuration and status....38
9.8 Monitor status....38
9.9 Partition router....39
9.10 Set input configuration....39
9.11 Set output configuration....40
9.12 Monitor configuration....42
9.13 Set environment alarm limits....44
9.14 List environment status....44
Appendix A – Supported commands....45

1 Introduction

This document describes the communication protocol for controlling Nevion products. Nevion VikinX Modular, Sublime and Compact routers in addition to Flashlink frames can be connected to Nevion Control Panels or computers through Ethernet, RS232 or RS422. As the protocol is ASCII based, changes can be made using a terminal program.

This document contains a command reference of the Modular Routing Protocol. All syntax in this document is presented in the Backus-Naur form (BNF).

2 Definitions

2.1 Routers and levels

Physical routers must be partitioned into one or more levels to make them controllable. Level is a logical router inside a physical router. A physical router output can not be included in more than one level. Input and output sequences are continuous and span from 0 to size-1. Refer to the Nevion Configurator user manual for more details.

Nevion products can only be partitioned into levels when Multicon is used. Without Multicon, the level size is equal to the physical router size.

2.2 Clients and servers

Typically, control panels are clients connecting to a server. Units controlling a router or Multicon system are clients. Servers are typically Sublime routers and the part of Multicon communicating with control panels. Units being controlled by others are servers.

2.3 Message format and structure

All message and command data consists of ASCII characters.

There are three types of messages:

- Commands.
- Command responses. Responses that the server sends back after receiving a command. Command responses are prefixed with a '?' character, may contain an echo of the command and are sent only to the client issuing the command.
- Status messages. Messages that the server sends to inform about status changes. Status messages are prefixed with a % character, do not echo any commands, and are sent to all connected clients.

A message consists of a sequence of ASCII characters (a string), terminated by two linefeed (value 0x10) characters. For the rest of this document, <LF> will represent a linefeed. Message termination is not shown in syntax descriptions for clarity. Literal strings are enclosed in single quotes in this document.

No part of the message will be executed before the complete message is received. The optional CRC is placed immediately preceding the pair of linefeed characters, see chapter 2.5 and 4.2. For commands, the CRC should be added after the last character of the command, before the pair of linefeeds. For command responses and status messages the CRC is preceded with a linefeed:

Command message:

```
<message> ::= <command> ['*'<CRC>]<LF><LF>
```

Command response:

```
<message> ::= '?' ['"'<command>'"']<LF><response>
[<LF>'*'<CRC>]<LF><LF>
```

Status message:

```
<message> ::= '%'<LF><status>[<LF>'*'<CRC>]<LF><LF>
```

Example:

```
z 13 52 27 *53A2
```

```
? "z 13 52 27"
ERROR: Unknown command
*61AD
```

Start and end of a text string is by means of a single " character. If a string contains a " or \ character, it must be preceded by a \ (like in the C programming language). Description fields may be up to 127 bytes long. UTF-8 is used for character encoding throughout the system.

2.4 Common syntax definitions

These syntax definitions are common to many functions.

```
<tables> ::= 'l'<levels>|'vt'<vtables>|'sg'<salvo_groups>

<levels> ::= <level>|<level>','<levels>|<level>'-'<level>

<vtables> ::= <vtable>|<vtable>','<vtables>|<vtable>'-'<vtable>

<salvo_groups> ::=
<salvo_group>|<salvo_group>','<salvo_groups>|<salvo_group>'-
'<salvo_group>

<param_groups> ::=
<param_group>|<param_group>','<param_groups>|<param_group>'-
'<param_group>

<vlevels> ::= <vlevel>|<vlevel>','<vlevels>|<vlevel>'-'<vlevel>

<salvos> ::= <salvo>|<salvo>','<salvos>|<salvo>'-'<salvo>
```

Mnemonics are supported for a lot of items. The alias and name is freely assignable, and will in many cases be called a label. The initial item name, alias and description is set during configuration. The description is not available for modification through this protocol.

```
<name_desc> ::= '"'<name>'"' '"'<alias>'"' '"'<description>'"'
                '"'<icon>'"'

<format> ::= 'TGSDI'|'HDSDI'|'SDI'|'AnalogHD'|
                'AnalogSD'|'AnalogAudio'|
                'AES'|'Audio'|'Data'|'Video'
```

2.5 Locking

LOCK CROSSPOINT

A lock is a property applied to an output, destination, salvo or parameter that makes that output, destination, salvo or parameter unchangeable to all users, including the user that locked it.

UNLOCK CROSSPOINT

Unlock is only effective if it is sent by a user with higher access level than the user which sent the lock or protect command, or the unlock is sent by the same user that sent the lock or protect command.

PROTECT CROSSPOINT

A protect is a property applied to an output, destination or salvo that makes that output, destination or salvo unchangeable to all other users, except for the one that protected it.

Refer to the Nevion Configurator for user management details.

The user that owns a locked or protected item is allowed to change the lock/protect state of this item. A user with higher access level then the user that owns the locked/protected item is allowed to change the state. An administrator can always unlock.

Lock command/state	Partial	Protect	Lock	Decimal
Unlock/unprotect	0	0	0	0
Lock	0	0	1	1
Protect	0	1	0	2
N/A	0	1	1	3
N/A	1	0	0	4
Partial lock	1	0	1	5
Partial protect	1	1	0	6
Partial lock + protect	1	1	1	7

Table 1. Lock command/state definitions

`<user ID>` is a two-byte number specifying the user id of the user that locked the item. User privileges are defined at configuration time. The optional user id is ignored in commands, but legal for backwards compatibility. Refer to chapter 3.2.

When setting a lock, only LOCK, PROTECT and UNLOCK are valid commands. In status messages, the state values are considered bitmask values, enabling a combined status value ranging from 0-7 (Table 1).

2.6 Check Sum (Cyclic Redundancy Checksum, CRC)

If the message is terminated by a * followed by 4 hexadecimal numbers, and then two <LF>, the message contains a check sum. The receiver of the message should verify that the check sum is correct, before executing the command.

The checksum is not needed for operation of the server. The checksum is a 16-bit word. The polynomial used for CRC calculation is defined in CCITT X.25 and UIT V.41. [*CRC] denotes proper placement of the checksum in commands and responses.

Example:

```
x 13 52 27 *53A2
```

2.7 Protocol error handling

Unknown or misspelled commands and syntactic errors return the same error message.

Example command response:

```
? "portocol"
ERROR: Unknown command
```

2.8 Future expansion

New commands and status messages may be added in the future, and messages may have parameters added. A client or host implementation must be able to ignore these extra parameters and messages without system failure. New parameters will always be added at the end of the command or status line.

3 Initial communication

3.1 Connection to server

Access to the server(s) for communication purposes is achieved by connecting either the serial port of your computer and/or by using an Ethernet connection (TCP at port 4381).

3.2 User identification and privileges

A default user will be set when a connection is opened. Access rights are set in relation to this. To change user to get additional privileges, use the login command. This command is new to MRP 3.0. Omitting username will return an ok response including the current username.

Logging in with the login command will set syntax to v3 (see 4.1).

```
<command> ::= 'login' [<username> [<password>]]
<response> ::= 'login' <username> <result>
<result> ::= 'ok'|'failed' [<error code>]
```

Example:

Command:	Response:
`login admin password`	`? "login admin password"` `login admin ok`

3.3 User list

Status commands only return a numerical value for identifying users, to map numerical values to user names, use this command.

```
<command> ::= 'userlist'
<response> ::= ['user' <username> <user ID> {<LF>'user' <username> <user ID>}]
```

Example:

Command:	Response:

`userlist`	`? "userlist"`
	`user admin 1`
	`user guest 2`
	`user Oper1 3`
	`user Oper2 4`
	`user Studio3 5`

4 Protocol configuration

4.1 Protocol syntax

Set the syntax used in the protocol. The syntax is local for each connection.

```
<command> ::= 'syntax' <mode>
<response> ::= <mode> 'mode'
```

Possible modes:

'compact'	When using the compact syntax, some of the syntax in Compact Router Control Protocol is used. This includes using input number 127 as disconnect commands and status on levels which support disconnect.
'v128'	When using the v128 syntax, disconnect commands and status is using the letter 'd'. This makes it possible to support data-routers with 128 or more inputs.
'v3'	Updated protocol syntax, where locking and user information is produced at all times. Turns on output of notifications not present in earlier versions, i.e. parameters, salvos, virtual routing updates etc.

Example:

Command:	Response:
syntax v128	? "syntax v128" v128 mode

The default syntax is v128. Always send the syntax command after connecting to the server. It is recommended to use v3 for new implementations; the others are for backwards compatibility only.

4.2 CRC in response

Switch on or off CRC in responses from the server. This command is local for each connection. Command response is empty apart from CRC and command echo.

```
<command> ::= 'crc' 'on'|'off'
<response> ::= ''
```

CRC in response is off by default.

Example:

Command:	Response:
crc on	? "crc on" *CBCE
ping *2E23	? "ping *2E23" pong *BD8C

4.3 Command echo

Switch on or off echo on commands. This command is local for each connection to the server. Command response is empty apart from CRC and command echo.

```
<command> ::= 'command_echo' 'on'|'off'

<response> ::= ''
```

Command echo is on by default.

Command:	Response:
command_echo off	? "command_echo off"

4.4 Protocol version

This command is used to determine the current protocol implementation. This command is new to MRP 3.0.

```
<command> ::= 'protocol'

<response> ::= <protocol description>
```

Response example:

```
? "protocol"

MRP rev 3.0
```

4.5 Server Ping

This command is used to check if the server is still responding.

```
<command> ::= 'ping'

<response> ::= 'pong'
```

Example:

Command:	Response:
ping	? "ping" pong

4.6 Status message suppression

To minimize load on clients, only physical crosspoint status messages are sent until a request has been made for more. Not specifying on or off will return the current setting.

```
<command> ::= 'filter' <filters> ['on'|'off']

<filters> ::= <filter> {',' <filter>}

<filter> ::=
'xpt'|'vxpt'|'salvo'|'param'|'status'|'ssp'|'name'|'event'


<response> ::= <filter_info> {<LF><filter_info>}

<filter_info> ::= 'filter' <filter> 'on'|'off'
```

Please note the sense of the filter setting; 'filter vxpt on' means allow virtual routing status across this interface.

The following table shows how the filter features and properties are connected together. The properties (status, ssp, name) turns on/off columns. The features (xpt, vxpt, salvo, param, event) turns on/off rows.

Filters	status	ssp	name
xpt	x, xl	sspo, sspi	l, in, out
vxpt	x	sspo, sspi	vt, vl, vcat, vsrc, vdest
salvo	x	sspi	sg, salvo
param	x	-	pg, par
event	event	-	-

5 Logical system structure

5.1 Level list

This command is used to get the list of all levels.

```
<command> ::= 'llist'
<response> ::= [<level_info> {<LF><level_info>}]

<level_info> ::= 'l'<level> <size> <format> <name_desc>
<format> ::= 'Video'|'Video-SDI'|'Video-HD-SDI'|'Video-3GHD-SDI'|'Video-Analogue'|'Audio'|'Audio-AES3'|'Audio-Analogue'|'Data'|'Data-Ported'
```

> Some product might also report the formats 'Data (Video)' and 'Data (Audio)', but these formats are not supported by revision 3 and later of this protocol.

Example:

Command:	Response:
llist	? "llist" l1 128x128 Video "Main router" "MAIN" "" "" l2 128x4 Video "Monitor router" "MONITOR" "" "" 16 8x8 Audio "Sound router" "AUDIO" "" "" l101 16x16 Data "Data router" "DATA" "" ""

5.2 Virtual Table list

This command is used to get the list of all virtual tables.

```
<command> ::= 'vtlist'
<response> ::= [<vtable_info> {<LF><vtable_info>}]
<vtable_info> ::= 'vt'<vtable> <n_srcs>'x'<n_dests>'x'<n_vlevels> <name_desc>
```

Example:

Command:	Response:
vtlist	? "vtlist" vt1 128x128x2 "Main A" "Main audio & video" "MAIN A" "" vt2 128x4x2 "Mon A" "" "MON A" "" vt16 16x8x4 "Edit 1" "Edit suite 1 routing" "EDIT 1" ""

5.3 Virtual Level list

This command is used to get the list of all virtual levels for a virtual table.

```
<command> ::= 'vllist' 'vt'<vtable>
<response> ::= [<vlevel_info> {<LF><vlevel_info>}]

<vlevel_info> ::= 'vl' 'vt'<vtable> <vlevel> <format> <name_desc>
<vlevel_ix>
```

Example:

Command:	Response:
vllist vt1	? "vllist vt1" vl vt1 1 "HDSDI" "Video" "Main router video" "VIDEO" "IconV" 3 vl vt1 2 "" "Audio" "Main router audio" "AUDIO" "IconA" 1

5.4 Parameter group list

This command is used to get the list of all parameter groups.

```
<command> ::= 'pglist'
<response> ::= [<pgroup_info> {<LF><pgroup_info>}]

<pgroup_info> ::= 'pg'<param_group> <size> <name_desc>
```

Example:

Command:	Response:
pglist	? "pglist" pg1 5 "Changeover controls" "Description1" "CHO 1" "" pg2 26 "Tally control lines" "Description2" "Tally" "" pg3 l2 "ADC-SDI-CC controls" "Description3" "ADC 1" ""

5.5 Salvo group list

This command is used to get the list of all salvo groups.

```
<command> ::= 'sglist'
<response> ::= [<sgroup_info> {<LF><sgroup_info>}]
<sgroup_info> ::= 'sg'<salvo_group> <size> <name_desc>
```

Example:

Command:	Response:
sglist	? "sglist" sg1 5 "Control room 1" "Control room 1 presets" "CTRL 1" "" sg2 26 "Edit suite 1" "Edit suite 1 presets" "EDIT 1" "" sg3 l2 "Control room 2" "Control room 2 presets" "CTRL 2" ""

5.6 Level input information list

This command is used to get the list of all inputs for a level.

> Inputs are numbered 0 to n-1, where n is the number of inputs!

```
<command> ::= 'inlist' 'l'<level>

<response> ::= [<input_info> {<LF><input_info>}]
<input_info> ::= 'in' 'l'<level> <n> <name_desc>
```

Example:

Command:	Response:
inlist l3	? "inlist l3" in l3 0 "VTR1" "Betacam player 1" "VTR1" "" in l3 1 "VTR2" "Betacam player 2" "VTR2" "" in l3 2 "VTR3" "Betacam recorder 1" "VTR3" "" in l3 3 "VTR4" "Betacam player 3" "VTR4" "" in l3 4 "GFX1" "Titler 1" "GFX1" "" in l3 5 "Chromakey" "Chromakey" "C-KEY" "" in l3 6 "AUX1" "Spare" "AUX1" "" in l3 7 "AUX2" "Spare" "AUX2" ""

5.7 Level output information list

This command is used to get the list of all outputs for a level.

> Outputs are numbered 0 to n-1, where n is the number of outputs!

```
<command> ::= 'outlist' 'l'<level>
<response> ::= [<output_info> {<LF><output_info>}]

<output_info> ::= 'out' 'l'<level> <n> <name_desc>
```

Example:

Command:	Response:
outlist l3	? "outlist l3" out l3 0 "C-KEY2" "Chromakey 2" "Chromakey input 2" "" out l3 1 "AUX1" "AUX1" "Spare" "" out l3 2 "VTR3" "VTR3" "Betacam recorder 1" "" out l3 3 "AUX2" "AUX2" "Spare" "" out l3 4 "GFX1" "GFX1" "Titler 1" "" out l3 5 "C-KEY1" "Chromakey 1" "Chromakey input 1" "" out l3 6 "PGM" "Program" "Program monitor" "" out l3 7 "PRE" "Preview" "Preview monitor" ""

5.8 Virtual sources information list

This command is used to get the list of all sources for a virtual table.

<command> ::= 'vsrclist' 'vt'<vtable>

<response> ::= [<vsrc_info> {<LF><vsrc_info>}]

<vsrc_info> ::= 'vsrc' 'vt'<vtable> <n> <name_desc> <src_ix> [<category_defs>]

<category_defs> ::= <category_id> ':' <cat_ix>[','<category_defs>]

Example:

Command:	Response:
vsrclist vt1	? "vsrclist vt1" vsrc vt1 0 "CAMERA 1" "Camera position 1" "CAM 1" "" 1 1:6 vsrc vt1 1 "CAMERA 1" "Camera position 2" "CAM 2" "" 2 1:1 vsrc vt1 2 "CAMERA 3" "Camera position 3" "CAM 3" "" 3 1:2 vsrc vt1 4 "CAMERA 4" "Camera position 4" "CAM 4" "" 4 1:4 vsrc vt1 5 "CAMERA 5" "Camera position 5" "CAM 5" "" 5 1:3 vsrc vt1 6 "CAMERA 6" "Camera position 6" "CAM 6" "" 6 1:5 vsrc vt1 8 "CAMERA 7" "Camera position 7" "CAM 7" "" 7 1:7 vsrc vt1 9 "SLOW MO 1" "Slow motion test" "SLO 1" "" 0 1:8,2:0

5.9 Virtual destinations information list

This command is used to get the list of all destinations for a virtual table.

```
<command> ::= 'vdestlist' 'vt'<vtable>
<response> ::= [<vdest_info> {<LF><vdest_info>}]

<vdest_info>::= 'vdest' 'vt'<vtable> <n> <name_desc> <dest_ix>
[<category_defs>]
```

Example:

Command:	Response:
vdestlist vt1	? "vdestlist vt1" vdest vt1 1 "MONITOR 1" "Top left monitor" "MON1" "" 0 3:1 vdest vt1 3 "MONITOR 2" "Top right monitor" "MON2" "" 1 3:4 vdest vt1 4 "MONITOR 3" "Bottom left monitor" "MON3" "" 2 3:2 vdest vt1 5 "MONITOR 4" "Bottom right monitor" "MON4" "" 3 3:3 vdest vt1 6 "LINK 1" "Link send 1" "LNK1" "" 4 4:1 vdest vt1 7 "LINK 2" "Link send 2" "LNK2" "" 5 4:2 vdest vt1 8 "LINK 3" "Link send 3" "LNK3" "" 6 4:3 vdest vt1 9 "PROGRAM" "Program output" "PRGM" "" 7 3:5,4:4

5.10 Category list

This command is used to get the list of all categories in a virtual table.

```
<command> ::= 'catlist' 'vt'<vtable>
<response> ::= [<cat_info> {<LF><cat_info>}]

<vdest_info>::= 'cat' 'vt'<vtable> <n> <name_desc>
```

Example:

Command:	Response:
catlist vt1	? "catlist vt1" cat vt1 1 "Cameras" "CAM" "Cameras in studio 1" "" cat vt1 2 "Monitors" "MON" "Monitors in MCR" ""

5.11 Salvo list

This command is used to get the list of all salvos in a salvo group.

```
<command> ::= 'salvolist' 'sg'<salvo_group>
<response> ::= [<salvo_info> {<LF><salvo_info>}]
<salvo_info> ::= 'salvo' 'sg'<salvo_group> <n> <name_desc>
```

Example:

Command:	Response:
salvolist sg1	? "salvolist sg1" salvo sg1 0 "Monitors" "Monitor wall" "MONs" "" salvo sg1 1 "Cameras" "Camera setup" "CAM" "" salvo sg1 2 "Editing" "Machine control routing" "EDIT" "" salvo sg1 3 "Oslo link" "Contribution from Oslo" "OSL" "" salvo sg1 4 "Bergen link" "Contribution from Bergen" "BGN" ""

5.12 Listing available parameters

List all parameters in a parameter group, analogue to levels for crosspoints.

```
<command> ::= 'parlist' 'pg'<param_group>
<response> ::= [<param_info> {<LF><param_info>}]

<param_info> ::= 'par' 'pg'<param_group> <n> <param_def>
[<overflow_def>] <read_def> <name_desc>
<param_def> ::= <enum_def>|<num_def>|<bool_def>
<enum_def> ::= 'enum' '"'<value>'"'{',' '"'<value>'"'}
<bool_def> ::= 'bool'
<num_def> ::= 'fixedpoint' <decimals> ['s'<stepsize>] 'u'<u_limit>
'l'<l_limit> ['d' '"'<unit>'"']
<read_def> ::= 'rw'|'r'
<overflow_def> ::= 'trunc'|'wrap'
```

<n>	Parameter index (in group). May be an arbitrary value in some implementation, do not assume it to be sequential.
<value>	Name/description of selectable option
<step_size>	All values are a multiple of this increment. Assumed to be 1.0 if not specified
<u_limit>	Upper limit for value
<l_limit>	Lower limit for value
<rw>	Value is read-write
<r>	Value is read-only
trunc	Leaves value at upper or lower limit if a value or increment outside permitted range is attempted.
wrap	Continues for remainder of increment from opposite limit if upper or lower limit is reached. Typical use: phase adjustment in degrees.
<unit>	Unit (dimension) of variable

Example:

Command:	Response:
parlist pg23	? "parlist pg23" par pg23 0 bool wrap rw "Laser" "Laser control for maintenance" "Laser" "" par pg23 1 enum "4:3","16:9LB","16:9" wrap rw "Aspect ratio" "Aspect ratio for output" "ARC" "" par pg23 2 fixedpoint 1 s1 u300 l-150 d"dB" trunc r "Video gain" "Analog Component video gain adjustment" "Gain" ""

For the enum data type, a simple integer is used for status and set messages, numbered from 0 to option count - 1 where n is the number of possible values.

5.13 Changing the alias of an item

All items that have a <name_desc> attached may have their alias (label) changed in real time. The response is the details of the item as returned by other commands. Multiple items of the same type can be changed by the same command. The multiple item features should be used for increasing performance when changing many aliases.

This command changes the system configuration. The complete system must be available to get a consistent configuration.

```
<command> ::= 'alias' <item> '"'<alias>'"' {<item> '"'<alias>'"'}
<response> ::= <item_info>

<item> ::= 'l'<level> ['in'<input>|'out'<output>]|
           'pg'<param_group> ['par'<param>]|
           'vt'<vtable> ['vl'<vlevel>|'cat'<cat>|
                'vsrc'<vsrc>|'vdest'<vdest>]|
           'sg'<salvo_group> ['salvo'<salvo>]
<item_info> ::= [<level_info> {<LF><level_info>}]|
           [<input_info> {<LF><input_info>}]|
           [<output_info> {<LF><output_info>}]|
           [<param_info> {<LF><param_info>}]|
           [<pgroup_info> {<LF><pgroup_info>}]|
           [<param_info> {<LF><param_info>}]|
           [<vtable_info> {<LF><vtable_info>}]|
           [<vlevel_info> {<LF><vlevel_info>}]|
           [<cat_info> {<LF><cat_info>}]
           [<vsrc_info> {<LF><vsrc_info>}]|
           [<vdest_info> {<LF><vdest_info>}]|
           [<sgroup_info> {<LF><sgroup_info>}]|
           [<salvo_info> {<LF><salvo_info>}]
```

Command:	Response:
alias l3 "L_CH"	% l3 8x8 Audio "Lvl3" "L_CH" "" ""
alias vt1 vl4 "RS422"	% vl vt1 4 Video-SDI "Ctrl" "RS422" "" "" 3
alias l3 in1 "COMP"	% in l3 1 "IN002" "COMP" "" ""
alias vt1 vdest5 "MON5"	% vdest vt1 5 "VDST005" "MON5" "" "" 8

5.14 Changing the name of an item

All items that have a <name_desc> attached may have their name changed in real time. The response is the details of the item as returned by other commands. Multiple items of the same type can be changed by the same command. The multiple item features should be used for performance when changing many names.

> This command changes the system configuration. The complete system must be available to get a consistent configuration.

<command> ::= 'name' <item> '"'<name>'"' {<item> '"'<name>'"'}

<response> ::= <item_info>

Refer to 5.13 for description of <item> and <item_info>.

Command:	Response:
name l3 "LeftChannel"	% l3 8x8 Audio "LeftChannel" "L_CH" "" ""
name vt1 vl4 "Control"	% vl vt1 4 Data "Control" "RS422" "" "" 3
name l3 in1 "Composite"	% in l3 1 "Composite" "COMP" "" ""
name vt1 vdest5 "Monitor5"	% vdest vt1 5 "Monitor5" "MON5" "" "" 8

5.15 Changing the description of an item

All items that have a <name_desc> attached may have their description changed in real time. The response is the details of the item as returned by other commands. Multiple items of the same type can be changed by the same command. The multiple item features should be used for performance when changing many names.

> This command changes the system configuration. The complete system must be available to get a consistent configuration.

<command> ::= 'desc' <item> '"'<desc>'"' {<item> '"'<desc>'"'}

<response> ::= <item_info>

Refer to 5.13 for description of <item> and <item_info>.

Command:	Response:
desc l3 "Main router"	% l3 64x64 3GHD-SDI "Level 1" "L1" "Main router" ""
desc l3 in1 "Camera in left corner"	% in l3 1 "Camera1" "Cam1" "Camera in left corner" ""
desc vt1 vdest5 "Monitor top-right wall"	% vdest vt1 5 "Monitor5" "MON5" "Monitor top-right wall" "" 8

6 System status

Status messages are sent for all items (crosspoints, virtual crosspoints, salvos and parameters) when the status has changed.

```
<status> ::= 'x' <status_item> <lock state> <user ID>
<status_item> ::=
      <s_xpt_item>|<s_vxpt_item>|<s_salvo_item>|<s_param_item>
<s_xpt_item> ::= 'l'<level> <input> <output>
<s_vxpt_item> ::= 'vt'<vtable> <source> <destination>
<s_salvo_item> ::= 'sg'<salvo_group> <salvo> <salvo_status>
<s_param_item> ::= 'pg'<param_group> <param> <value>
```

6.1 Status messages

Whenever the state of a monitored item, i.e. crosspoints, parameters or signal presence, in the system changes, status messages are sent to all listeners. A single state change may generate any number of status messages, caused by the interconnection of parameters, physical and virtual crosspoints and the use of salvos.

6.2 Request Crosspoint Status — Entire Level

This command is used to requests the crosspoint status of all destinations on a specified level. For compact and v128 modes, lock state and user ID are omitted, and the `sl` command is defined.

```
<command> ::= 's' 'l'<level>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> [<lock state> <user ID>]
<item> ::= 'l'<level> <input> <output>

<command> ::= 'sl' 'l'<level> (v128 and compact modes only)
<status> ::= 'xl' <item> <lock state> <user ID>  (v128/compact modes)
```

Example:

Command:	Response:

`s l16`	`? "s l16"`
	`x l16 0 0 0 500`
	`x l16 1 1 0 500`
	`x l16 1 2 0 500`
	`x l16 3 3 0 500`
	`x l16 2 4 1 750`
	`x l16 5 5 1 750`
	`x l16 7 6 5 500`
	`x l16 7 7 5 500`

6.3 Request Crosspoint Status – Entire Level – Compact

This command requests the crosspoint status of a specific level. This will give a response in a compact format.

```
<command> ::= 'r' 'l'<level>
<response> ::= [<level_status> {<LF><level_status>}]
<level_status> ::= 's' 'l'<level> {<input>}
```

Example:

Command:	Response:
r l16	? "r l16" s l16 0 1 1 3 2 5 7 7

6.4 Request Virtual Crosspoint Status — Entire Virtual Table

This command is used to request the crosspoint status of all destinations in a specified virtual table. 'u' (unknown) is sent if the status cannot be determined, e.g. due to disconnection of a router.

"Cross-level break-away" routing is indicated similarly to ordinary break-away routing, this can greatly reduce the number of required sources if operational flexibility is needed with respect to separation of inputs into separate virtual levels, depending on intended purpose.

```
<command> ::= 's' 'vt'<vtable>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> <lock state> <user ID>
<item> ::= 'vt'<vtable> <vsrc_item> <vdest>
<vsrc_item> ::= <vsrc>|'u'|<vsrc_list>
<vsrc_list> ::= ('vl'<vlevel> <vsrc>|'u' <vlevel lock state>
<vlevel user>)+
```

Example:

Command:	Response:
s vt4	? "s vt4" x vt4 3 0 0 500 x vt4 u 1 0 500 x vt4 3 2 0 500 x vt4 vl1 4 0 500 vl2 2 0 500 3 0 500 x vt4 vl1 4 1 501 vl2 4 2 500 4 7 500 x vt4 2 5 0 500 x vt4 vl1 2 0 500 vl2 4 0 500 6 0 500 x vt4 6 7 0 500

6.5 Request Salvo Status — Entire Salvo Group

This command is used to requests the status of all salvos in a specified salvo group.

```
<command> ::= 's' 'sg'<salvo_group>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> <lock state> <user ID>
<item> ::= 'sg'<salvo_group> <connection_status> <salvo>
<connection_status> ::= '1'|'0'
```

<connection_status>	Current status for the salvo, can be one of the following values:	
	1	Fully connected.
	0	One or more crosspoints or parameters do not match the salvo

Example:

Command:	Response:
s sg1	? "s sg1" x sg1 1 1 0 3 x sg1 2 1 0 3 x sg1 3 0 0 3 x sg1 4 0 0 3 x sg1 5 1 0 3

6.6 Request status – partial status

This command is used to retrieve partial status of levels, virtual tables and salvos.

6.6.1 Levels

```
<command> ::= 'si' 'l'<level> <range>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> [<lock state> <user ID>]
<item> ::= 'l'<level> <input> <output>
```

<range>	Range of outputs to retrieve status from, can be only one value or a range of values from min to max.

Example:

Command:	Response:
si l2 3-5	? "si l2 3-5" x l2 1 3 0 3 x l2 8 4 0 3 x l2 4 5 0 3
si l3 124	? "si l3 124" x l3 54 124 0 3

6.6.2 Virtual router

```
<command> ::= 'si' 'vt'<vtable> <range>

<response> ::= [<status> {<LF><status>}]

<status> ::= 'x' <item> <lock state> <user ID>

<item> ::= 'vt'<vtable> <vsrc_item> <vdest>

<vsrc_item> ::= <vsrc>|'u'|<vsrc_list>

<vsrc_list> ::= ('vl'<vlevel> <vsrc>|'u' <vlevel lock state>
<vlevel user>)+
```

<range>	Range of virtual destinations to retrieve status from, can be only one value or a range of values from min to max.

Example:

Command:	Response:
si vt2 4-6	? "si vt2 4-6" x vt2 3 4 0 3 x vt2 17 5 0 3 x vt2 4 6 0 3
si vt3 14	? "si vt3 14" x vt3 10 14 0 3

6.6.3 Salvos

```
<command> ::= 'si' 'sg'<salvo_group> <range>

<response> ::= [<status> {<LF><status>}]

<status> ::= 'x' <item> <lock state> <user ID>

<item> ::= 'sg'<salvo_group> <connection_status> <salvo>

<connection_status> ::= '1'|'0'
```

<range>	Range of virtual destinations to retrieve status from, can be only one value or a range of values from min to max.

Example:

Command:	Response:
si sg2 4-6	? "si sg2 4-6" x sg2 4 1 0 3 x sg2 5 1 0 3 x sg2 6 0 0 3
si sg3 14	? "si sg3 14" x vt3 14 1 0 3

6.6.4 Parameters

```
<command> ::= 'si' 'pg'<param_group> <range>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> [<lock state> <user ID>]
<item> ::= 'pg'<param_group> <param> <value>
<value> ::= <integer value> | 'u'
```

<range>	Range of parameters to retrieve status from, can be only one value or a range of values from min to max.

Example:

Command:	Response:
si pg2 3-5	? "si pg2 3-5" x pg2 8 3 0 3 x pg2 1 4 0 3 x pg2 3 5 0 3
si pg7 29	? "si pg7 29" x pg7 93 29 0 3

6.7 Input signal presence status

This command and status applies to levels, virtual tables and salvos.

Input signal presence messages are also sent as status messages when the status changes.

```
<command> ::= 'sspi' <tables>

<response> ::= [<input_status> {<LF><input_status>}]


<input_status> ::= 'sspi' <input_item>|<source_item>|<salvo_item>

<input_item> ::= 'l'<level> <input> <presence>

<source_item> ::= 'vt'<vtable> <source> <presence>|('vl'<vlevel>
<presence>)+

<salvo_item> ::= 'sg'<salvo_group> <salvo> <presence>

<presence> ::= 'p'|'u'|'m'
```

<presence>	Current status for the input, source or salvo, can be one of the following values:	
	p	Present. All signals are reported as present.
	m	Missing. One or more inputs do not have a valid signal
	u	Unknown. One or more inputs are not able to report if a signal is present or not.

Examples:

Command:	Response:		Command:	Response:
sspi l16	? "sspi l16" sspi l16 0 p sspi l16 1 m sspi l16 2 m sspi l16 3 m sspi l16 4 p sspi l16 5 p sspi l16 6 p sspi l16 7 p		sspi sg1	? "sspi sg1" sspi sg1 1 p sspi sg1 2 u sspi sg1 3 m sspi sg1 4 u sspi sg1 5 u
sspi vt4	? "sspi vt4" sspi vt4 0 p sspi vt4 1 vl1 m vl2 u sspi vt4 2 m sspi vt4 3 u sspi vt4 4 p sspi vt4 5 u sspi vt4 6 vl1 p vl2 m sspi vt4 7 vl1 p vl2 u			

6.8 Output signal presence status

This command and status applies to levels and virtual tables.

Output signal presence messages are also sent as status messages when the status changes.

```
<command> ::= 'sspo' <tables>
<response> ::= [<output_status> {<LF><output_status>}]

<output_status> ::= 'sspo' <output_item>|<dest_item>
<output_item> ::= 'l'<level> <output>  <presence>
<dest_item> ::= 'vt'<vtable> <destination> <presence>|('vl'<vlevel>
<presence>)+
<presence> ::= 'p'|'u'|'m'
```

<presense>	Current status for the output or destination, can be one of the following values:	
	p	Present. All signals are reported as present.
	m	Missing. One or more outputs do not have a valid signal
	u	Unknown. One or more outputs are not able to report if a signal is present or not.

Examples:

Command:	Response:		Command:	Response:
sspo l16	? "sspo l16" sspo l16 0 p sspo l16 1 m sspo l16 2 m sspo l16 3 m sspo l16 4 p sspo l16 5 p sspo l16 6 p sspo l16 7 p		sspo vt4	? "sspo vt4" sspo vt4 0 p sspo vt4 1 m sspo vt4 2 m sspo vt4 3 u sspo vt4 4 vl1 m vl2 p sspo vt4 5 vl1 p vl2 u sspo vt4 6 p sspo vt4 7 m

6.9 Getting status for all parameters in group

```
<command> ::= 's' 'pg'<param_group>
<response> ::= [<status> {<LF><status>}]
<status> ::= 'x' <item> <lock state> <user ID>
<item> ::= 'pg'<param_group> <param> <value>
<value> ::= <integer value> | 'u'
```

Examples:

Command:	Response:

`s pg23`	`? "s pg23"`
	`x pg23 0 1 0 500`
	`x pg23 1 u 0 500`
	`x pg23 2 135 0 500`

7 System operation commands

7.1 Crosspoint Take

This command directly takes the specified crosspoint. The command specifies which inputs to connect to which outputs and on what levels. Several crosspoint takes can be specified on a single command line. In such cases all specified crosspoints are taken simultaneously.

```
<command> ::= 'x' <crosspoint> {<crosspoint>}

<crosspoint> ::= 'l'<levels> <in_out_pair> {<in_out_pair>}
<in_out_pair> ::= <in> <outs>
<outs> ::= <out>|<out>','<outs>|<out>'-'<out>
```

A single Crosspoint Take command can switch multiple crosspoints. You will however receive one status message for each crosspoint that is switched.

<in>	Input number in the range from zero to one less than the number of inputs on the level. <in> can also be one of the following characters:	
	d	Disconnect. Used when a destination is not connected to any source.
	o	Out of range. Used when a destination is connected to a source not included in a level. This may occur with overlapping partitions. Only valid in the response.
	u	Undefined/Unknown. This might be used at start-up before the status is read from the physical router. It is also used on ported routers where a port can be used as an input or an output. Only valid in the response.
<out>	Output number in the range from zero to one less than the number of outputs on the level.	

Examples:

Command	Desired action
`x l3 52 27`	Switch input 53 to output 28 in level 3.
`x l1,2 2 4-6,8`	Switch input 3 to output 5, 6, 7 and 9 in levels 1 and 2.
`x l37 1 1 2 2`	In level 37 switch input 2 to output 2 and input 3 to output 3.

7.2 Order of execution

The following rules apply:

- For all crosspoints specified directly or indirectly through the use of virtual tables and salvos, the commands will be executed as soon as possible.
- All crosspoints residing in the same router will be switched at the same video line if this is supported by hardware.
- No further guaranties are given for time of execution.

7.3 Virtual Crosspoint Take

This command directly takes the specified virtual crosspoint. The command specifies which virtual sources to connect to which virtual destinations and in which virtual tables. Several virtual crosspoint takes can be specified on a single command line. In such cases all specified virtual crosspoints are taken simultaneously.

Virtual level break-away is handled by specifying which virtual levels of the virtual table are included in the command. If no virtual levels are specified, all virtual levels are switched.

```
<command> ::= 'x' 'vt'<vtables> <vcrosspoint> {<vcrosspoint>}


<vcrosspoint> ::= <vcrosspoint_std>|<vcrosspoint_xlevel>
<vcrosspoint_std> ::= ['vl'<vlevels>] <src_dest_pair>+
<src_dest_pair> ::= <src> <dests>
<vlevels> ::= <vlevel>|<vlevels>','<vlevels>|<vlevel>'-'<vlevel>
<dests> ::= <dest>|<dests>','<dests>|<dest>'-'<dest>


<vcrosspoint_xlevel> ::= 'vl'<dest vlevel>':vl'<src vlevel>
<src_dest_pair>+
```

Examples:

Command	Desired action
`x vt3 2 7`	Switch source 3 to destination 8 in virtual table 3.
`x vt1,2 2 4`	Switch source 3 to destination 5 in virtual tables 1 and 2.
`x vt7 vl2-4 1 1 2 2`	Switch source 2 to destination 2 and source 3 to destination 3 in virtual table 7, switching only virtual levels 2, 3 and 4
`x vt7 vl2:vl3 6 2`	"Cross-level break-away" Switch source 7 in virtual level 3 to destination 3 in virtual level 2 (assuming the levels are compatible).

Cross-level break-away is preliminary

7.4 Salvo Take

This command takes, reverts or stores the specified salvo. Store and restore are only valid if the salvo has enabled 'salvo restore'. Refer to salvo groups in the Nevion Configurator.

```
<command> ::= 'x' 'sg'<salvo_group> <salvos> ['s'|'r']
```

`s`	Means to store all current statuses for crosspoints, virtual crosspoints, and parameters in the salvo. This does not directly take. Storage is performed locally (not in the complete system) and is lost on reset. Refer to the salvo update command for permanent system storage.
`r`	Means to restore the previously stored salvo using direct take.

7.5 Setting parameter

Update a parameter with a new assigned value or an increment.

```
<command> ::= 'x' ('pg'<param_group> <param> ['rel'] <value>)+
```

rel	Means the indicated value is an increment to the current value

Examples:

Command:	Response:
x pg23 1 1	% x pg23 1 1 0 500
x pg23 2 rel -16	% x pg23 2 135 0 500

7.6 Lock/protect

User ID is ignored in syntax v3 mode, where it is implied by the connection.

```
<command> ::= 'ld' <lock_item> <lock command> [<user ID>]


<lock_item> ::=
<l_xpt_item>|<l_vxpt_item>|<l_salvo_item>|<l_param_item>
<l_xpt_item> ::= 'l'<level> <output>
<l_vxpt_item> ::= 'vt'<vtable> [<vlevels>] <destination>
<l_salvo_item> ::= 'sg'<salvo_group> <salvo>
<l_param_item> ::= 'pg'<param_group> <param>
```

7.7 Reset lock/protect

This command resets all locks and protects on the specified tables.

```
<command> ::= 'lr' <table>
```

7.8 Diagonal

This command sets the level in diagonal (input1 to output1, input2 to output2, ...)

```
<command> ::= 'diagonal' 'l'<level>
```

8 Server Alarms and Events

8.1 Alarm status

Alarms are sent as status messages asynchronously from the controller. Alarms are stored in the system until they are deactivated and removed.

```
<status> ::= <alarm>
<alarm> ::= 'alarm' <alarmid> <alarm_details> <alarm_desc>
<alarm_details> ::= <state> <time> <user ID> <severity>
<alarm_desc> ::= '"'<origin>'"' '"'<description>'"'
```

<alarmid>	Unique number identifying the alarm.	
<state>	State of the alarm:	
	0	Removed: the alarm is not present. The alist response is not required to include removed alarms, if acknowledge is supported. Acknowledge of a restored alarm will result in a removed response.
	1	New: the alarm is active and has not been acknowledged.
	2	Restored: the alarm has been active, but is now gone. Do acknowledge this alarm to remove it. Devices without acknowledge support will not respond with alarms in this state.
	3	Acknowledged: the alarm is active and has been acknowledged. The alarm will be removed if it becomes inactive. Devices without acknowledge support will not respond with alarms in this state. Do unacknowledge this alarm to set it to new.
<time>	Time when the alarm state was last changed. Seconds since January 1, 1970 (midnight UTC). Devices without a real-time clock can respond with zero.	
<severity>	The degree of alarm seriousness.	
	0	Ok
	1	Information
	2	Warning
	3	Minor
	4	Major
	5	Critical
	6	Unknown
<origin>	A description of the originator of the alarm. This string is comma separated with details from highest layers first and closest to the originating device last. Multiple layers can be used.	
<description>	A description of the alarm.	

8.2 Alarm list

Lists all alarms in the controller.

```
<command> ::= 'alist' [<severity]
<response> ::= [<alarm> {<LF><alarm>}]
```

See the alarm status for more information about the response.

<severity>	List all alarms with this severity or higher.

Example:

Command:	Response:
alist 1	? "alist 1" alarm 429 3 1324755414 3 5 "STUDIO3-SL-HD-1616" "Power B failed" alarm 302183 2 1381402151 3 2 "ROOM4-RACK2, Frame 2, Card 7, FRS-HD-CHO" "Sync source"

8.3 Alarm acknowledge

Change the state of alarms.

```
<command> ::= 'alarm' <alarms> <command>
<alarms> ::= <alarmid>|<alarmid>','<alarmid>|<alarmid>'-'<alarmid>
<response> ::= [<alarm>{<LF><alarm>}]
```

<command>	Action to perform on the alarm.	
	0	Unacknowledge. This will change alarms in the Acknowledged state to New. Unacknowledge of other states will be ignored.
	1	Acknowledge. This will change alarms in the Restored state to Removed or New to Acknowledged.

Examples:

Command:	Response:
alarm 429 0	% alarm 429 1 1381484524 3 5 "STUDIO3-SL-HD-1616" "Power B failed"
alarm 302183 1	% alarm 302183 0 1381484691 3 2 "ROOM4-RACK2, Frame 2, Card 7, FRS-HD-CHO" "Sync source"

8.4 Event status

Events are sent as status messages asynchronously from the controller. Events occur once and can be handled by connected clients.

```
<status> ::= 'event' <type> <event_state> <description>
```

<type>	A number representing the type of event that occured.	
	1000	Configuration change. This event will be emitted on major changes in the controller configuration. When a control-system receives this event, it is time to rescan the server system.
<event_state>	State of the event:	
	0	Inactive
	1	Active
<description>	Text without quotes describing the event.	

9 Deprecated commands

These commands are no longer used for new products.

9.1 Request Crosspoint Locking Status — Single Destination

This command requests the locking cross point status of a specific destination on a specific level.

Command

```
si l<level1> <dest1> [[l<level1>] <dest1>] [*CC]<LF><LF>
```

Response

```
%<LF>
xl l<level> <src> <dest> <lock> <group no><LF>
[xl l<level> <src> <dest> <lock> <group no><LF> …]
[*CC<LF>]
<LF>
```

9.2 Router list

This command is used to request a list over all routers in the system.

Command:

```
rlist<LF><LF>
```

Response:

```
? "rlist" <LF>
r<rno> <size> "<router description>"<LF>
[r<rno> …]
[*CC<LF>]
<LF>
```

9.3 Router partition list

This command is used to get the list of partitions of all routers in a system. A partition is the link between a router and a level.

Command

```
plist<LF><LF>
```

Response

```
? "plist" <LF>
r<rno> p<pno> l<level> "<description>" <lIn>:<pIn>
<lOut>:<pOut><LF>
[r<rno> p<pno> l<level> …]
[*CC<LF>]
<LF>
```

9.4 Bus list

A router is connected to the controller on some physical connection. This is called a bus in this protocol. A bus can have zero to many routers connected. This command lists each router in the controller and which bus they are connected to.

Command:

```
blist[*CC]<LF><LF>
```

Response:

```
? "blist"<LF>
<router_bus_relations>
[*CC<LF>]
<LF>
```

Where

```
<router_bus_relations>=<router_bus_relation>[
<router_bus_relations>]
<router_bus_relation>=r<rno> b<busno>[ {local|remote}[
<interface_type>][ <protocol>]<LF>
<interface_type>={internal|serial|tcpip[ <ip>:<port>]}
<protocol>={ncb|modular|triton|probel-sw-p-88|leitch-pass-through}
```

9.5 Module list

Some routers are built up by modules. This command lists all modules in this router.

Command:

```
mlist[*CC]<LF><LF>
```

Response:

```
? "mlist"<LF>
m<module> b<busno> <modulename>
```

9.6 List input configuration and status

Lists each input's configuration and status.

Command:

```
sin b<busno>[*CC]<LF><LF>
```

Response:

```
? "sin b<busno>"<LF>
[in b<busno> m<module> <input>
  [eq={unknown|enabled|bypass}]
  [signal={unknown|no|yes}]
  <LF>]
[in …]
[*CC<LF>]
<LF>
```

9.7 List output configuration and status

Lists each output's configuration and status.

Command:

```
sout b<busno>[*CC]<LF><LF>
```

Response:

```
? "sout b<busno>"<LF>
[out b<busno> m<module> <output>
  [rcl={enabled|locked|locked,autobypass|
        autobypass|bypass|mute}]
  [br=<bitrate>] [asi={enabled|disabled}]
  [sr={auto|sd|hd}]
  <LF>]
[out …]
[*CC<LF>]
<LF>
```

9.8 Monitor status

Show monitor status.

Command:

```
smonitor [b<busno> m<module> <monitor>][*CC]<LF><LF>
```

Response:

```
[monitor b<busno> m<module> <monitor>
  [type=edh [mask={none|all|[trs][[,]eav]
                           [[,]sav][[,]lnum]
                           [[,]lcrc][[,]ycrc]
                           [[,]ccrc][[,]ancs]
                           [[,]ycs][[,]ccs]
                           [[,]lock][[,]apcrc]
                           [[,]ffcrc][[,]vs]}]
            [error_count=<count>]
            [unlocked|locked] [asi] [sd|hd]
            [[y|c]data_format={sdti_dvcpro|
                               sdti_dvcpro_ecc|
                               sdti_dvcam|
                               sdti_cp|sdti_fixed|
                               sdti_variable|
                               sdi|dvb_asi|tdm|
                               unknown}]
```

```
            [standard={1280x720/60|1280x720/30|
                       1280x720/50|1280x720/25|
                       1280x720/24|1920x1080|
                       1920x1080/30|1920x1080|
                       1920x1080/25|1920x1080/24|
                       1920x1080/50|1920x1080/60|
                       525-line|625-line|hd|sd|unknown}]
    <LF>]
[monitor …]
[*CC<LF>]
<LF>
```

9.9 Partition router

This command is used for partition the router. See chapter 2.1 for more information about router partitions.

Command

```
partition r<rno> p<pno> l<level> "<desciption>" <lIn>:<pIn>
  <lOut>:<pOut> [p<pno> …][*CC]<LF><LF>
```

where:

<rno>	Router number
<pno>	Partition number
<level>	Level number
<description>	Description
<lIn>	Logical input number range
<pIn>	Physical input number range
<lOut>	Logical output number range
<pOut>	Physical output number range

Response:

```
? "<command>"<LF>
[*CC<LF>]
<LF>
```

9.10 Set input configuration

Set the input configuration.

Command:

```
in b<busno1> m<module1> <input1>
  [eq={enable|bypass}][*CC]<LF><LF>
```

Response:

```
? "<command>"<LF>
[in b<busno> m<module> <input>
  [eq={unknown|enabled|bypass}]
  [signal={unknown|no|yes}]
  <LF>]
[in …]
[*CC<LF>]
<LF>
```

Where:

<busno>	Bus number.	
<module>	Module number.	
<input>	Input number. Starts at 0 and increases to the router size – 1.	
eq	Equaliser configuration. This option is only available on VD128M and HD128M.	
	unknown	The input equalizer configuration is unknown.
	enabled	The input equalizer is enabled.
	bypass	The input is configured not to use the input equalizer.
signal	Reports if it is signal on the input.	
	unknown	This can occur on routers that don't have the ability to read this status. It can also occur at startup before the input status is read.
	no	It is no signal on this input.
	yes	It is signal on this input.

Example:

```
in b0 m0 5 eq=bypass
in b0 m0-1 0-31 eq=enable
in b0-2 m0-3 0-31 eq=bypass
```

9.11 Set output configuration

Set output configuration

Command:

```
out b<busno1> m<module1> <output1>
  [rcl={enable|bypass|autobypass|mute}]
  [asi={enable|disable}]
  [sr={auto|sd|hd}]
  [*CC]<LF><LF>
```

Response:

```
? "<command>"<LF>
[out b<busno> m<module> <output>
  [rcl={enabled|locked|locked,autobypass|
        autobypass|bypass|mute}]
  [br=<bitrate>]
  [asi={enabled|disabled}]
  [sr={auto|sd|hd}]
  <LF>]
[out ...]
[*CC<LF>]
<LF>
```

Where:

<busno>	Bus number.	
<module>	Module number.	
<output>	Output number. Starts at 0 and increases to the router size – 1.	
rcl	Reclocker configuration and status.	
	enabled	The output is configured to enable the reclocker, but it is not locked.
	locked	The output is configured to enable the reclocker, and it is locked.
	Locked, autobypass	The output is configured to bypass the reclocker when it's not locked. It is now locked.
	autobypass	The output is configured to bypass the reclocker when it's not locked, which it isn't.
	bypass	The output is configured to bypass the reclocker.
	mute	The output is configured to mute.
br	When the reclocker is locked this option is showing the bitrate it is locked to.	
asi	Prevents a false lock to 177Mb/s when using DVB/ASI.	
	enabled	Prevents the reclocker to lock at 177Mb/s.
	disabled	Makes it possible for the reclocker to lock at 177Mb/s.
sr	Shows the slew rate the cable driver is configured to use.	
	auto	The slew rate is selected automatically.
	sd	The output will meet the SMPTE 259M rise/fall time specifications.
	hd	The output will meet the SMPTE 292M rise/fall time specifications.

9.12 Monitor configuration

Configure the monitor matrix. This is only available on some crosspoint-boards.

Command:

```
monitor b<busno1> m<module1> <monitor1>
  [restart]
  [mask={none|all|[trs][[,]eav][[,]sav][[,]lnum]
                 [[,]lcrc][[,]ycrc][[,]ccrc]
                 [[,]ancs][[,]ycs][[,]ccs]
                 [[,]lock][[,]apcrc][[,]ffcrc][[,]vs]}]
  [*CC]<LF><LF>
```

Response:

```
? "<command>"<LF>
[monitor b<busno> m<module> <monitor>
  [type=edh [mask={none|all|[trs][[,]eav]
                           [[,]sav][[,]lnum]
                           [[,]lcrc][[,]ycrc]
                           [[,]ccrc][[,]ancs]
                           [[,]ycs][[,]ccs]
                           [[,]lock][[,]apcrc]
                           [[,]ffcrc][[,]vs]}]
            [error_count=<count>]
            [unlocked|locked] [asi] [sd|hd]
            [[y|c]data_format={sdti_dvcpro|
                               sdti_dvcpro_ecc|
                               sdti_dvcam|
                               sdti_cp|sdti_fixed|
                               sdti_variable|
                               sdi|dvb_asi|tdm|
                               unknown}]
            [standard={1280x720/60|1280x720/30|
                       1280x720/50|1280x720/25|
                       1280x720/24|1920x1080|
                       1920x1080/30|1920x1080|
                       1920x1080/25|1920x1080/24|
                       1920x1080/50|1920x1080/60|
                       525-line|625-line|hd|sd|unknown}]
    <LF>]
```

[monitor …]

[*CC<LF>]

<LF>

Where:

<busno>	Bus number.	
<module>	Module number.	
<monitor>	Monitor number.	
restart	Reset the error_count to zero.	
mask	EDH errors are counted in error_count. The mask option is used to select which error types that should be included in this count.	
	none	Don't count any error.
	all	Count all errors.
	trs	Count TRS errors. This is equal to enabling eav and sav.
	eav	Count end of active video errors.
	sav	Count start of active video errors.
	lnum	Count line number error flags.
	lcrc	Count CRC error flags. This is equal to enabling ycrc and ccrc.
	ycrc	Count luma CRC error flags.
	ccrc	Count chroma CRC error flags.
	ancs	Count checksum error flags. This is equal to enbaling ycs and ccs.
	ycs	Count luma checksum error flags.
	ccs	Count chroma checksum error flags.
	lock	Count lock error flags.
	apcrc	Count active picture CRC error flags.
	ffcrc	Count full field CRC error flags.
	vs	Count video standard error flags.
error_count	Errors counted since last restart. This is a 16 bits number. When it reach its limit, it will stop counting.	
ydata_format	Luma data format. Only availavle on HD signals.	
cdata_format	Chroma data format. Only available on HD signals.	
data_format	Data format. On HD signals ydata_format and cdata_format is used instead.	

9.13 Set environment alarm limits

Set environment alarm limits

```
<command> ::= 'env' 'b'<busno> 'm'<module> 'nom'|'ul'|'ll'
['temp='<value>]
['voltage1='<value>['mV'|'V']]['voltage2='<value>['mV'|'V']]
['voltage3='<value>['mV'|'V']]['voltage4='<value>['mV'|'V']]
['fan1='<value>['rpm']]['fan2='<value>['rpm']]
['timer1='<value>]['timer1='<value>]


<response> ::= [<env_info> {<LF><env_info>}]

<env_info> ::= 'env' 'b'<busno> 'm'<module> 'nom'|'ul'|'ll'
['temp='<value>]
['voltage1='<value>]['voltage2='<value>]
['voltage3='<value>]['voltage4='<value>]
['fan1='<value>]['fan2='<value>]
['timer1'=<value>]['timer1='<value>]
```

Where:

nom	List nominal values
ul	List upper limit
ll	List lower limit

```
env b1 m0 nom voltage1=1700mV


? "env b1 m0 nom voltage1=1700mV"

env b1 m0 nom voltage1=1700mV voltage2=3300mV voltage3=5000mV
voltage4=12000mV
```

9.14 List environment status

List the environment status.

```
<command> ::= 'senv' ['stat'|'nom'|'ul'|'ll']

<response> ::= [<env_info> {<LF><env_info>}]


senv


? "senv"

env b1 m0 temp=38.00 voltage1=1794mV voltage2=3222mV
voltage3=5002mV voltage4=11860mV fan1=5758rpm fan2=5756rpm
timer1=8864 timer2=8864

env b1 m1 temp=33.75 voltage1=1791mV voltage2=3193mV
voltage3=5024mV voltage4=12014mV fan1=3243rpm fan2=3239rpm
timer1=24657 timer2=24657
```

Appendix A – Supported commands

This appendix describes which commands are supported by devices developed by Nevion.

Command	Description	Minimum	Multicon	Sublime	Reference
login	User login		x	x	3.2
syntax	Protocol syntax	1	x	2	4.1
crc	CRC in response	3	x	x	4.2
command_echo	Command echo		x		4.3
protocol	Show current protocol implementation		x		4.4
ping	Server ping	x	x	x	4.5
filter	Status message suppression		x		4.6
llist	Level list	x	x	x	5.1
vtlist	Virtual table list		x		5.2
vllist	Virtual level list		x		5.3
pglist	Parameter group list		x		5.4
sglist	Salvo group list		x		5.5
inlist	Level input list		x		5.6
outlist	Level output list		x		5.7
vsrclist	Virtual source list		x		5.8
vdestlist	Virtual destination list		x		5.9
catlist	Category list		x		5.10
salvolist	Salvo list		x		5.11
parlist	Parameter list		x		5.12
alias	Set alias		x		5.13
name	Set name		x		5.14
desc	Set description		x		5.15
s l	Status level		x		6.2
r l	Status level, Compact format		x[4]		6.3
s vt	Status virtual crosspoint table		x		6.4
s sg	Status salvo group		x		6.5

1 Use v128 syntax
2 Uses v128 syntax
3 It is recommended to support CRC if the protocol is used on an interface which doesn't include error checking.
[4] For backwards compatibility only, do not use for new implementations

s pg	Status parameter group		x		6.9
si	Partial status (l, vt, sg, pg)		x		6.6
sspi	Status input signal presence		x	x	6.6
sspo	Status output signal presence		x	x	6.8
x l	Crosspoint take	x	x	x	7.1
x vl	Virtual crosspoint take		x		7.3
x s	Salvo take		x		7.4
x pg	Set parameter value		x		7.5
ld	Lock and protect crosspoint		x	x	7.6
lr	Reset all locks and protections on a level		x		7.7
alarm	Server alarms		x	x[5]	8.1
alist	Alarm list		x	x	8.2
event	Server events		x[6]	x[7]	8.4
sl	Status Locking – level		x[8]	x	6.2
rlist	Router list				9.2
plist	Router partition list				9.3
blist	Bus list				9.4
mlist	Module list				9.5
sin	List input configuration and status				9.6
sout	List output configuration and status				9.7
smonitor	Monitor status				9.8
partition	Partition router				9.9
in	Set input configuration				9.10
out	Set output configuration				9.11
monitor	Set monitor configuration				9.12
env	Set environment alarm limits				9.13
senv	List environment status				9.14

A server which is implementing this protocol must support all the commands in the minimum column.

A control system which supports this protocol must not require any of the commands not in the minimum column.

[5] Command not implemented, status returned on changes
[6] Command not implemented, status returned on changes
[7] Command not implemented, status returned on changes
[8] For backwards compatibility only, do not use for new implementations